

3. Syntaxe de Base

Jean-Luc Falcone

Août 2019

Déclarations

Il existe 4 manières de déclarer une variable:

```
x = 2; // 'x' est une variable globale
```

```
var y = 2; // 'y' est une variable locale à la fonction
```

```
let z = 2; // 'z' est une variable locale au bloc (ES6)
```

```
const PI = 3.14159; // 'PI' est une constante locale (ES6)
```

var vs. let

```
function varTest() {  
  var x = 1;  
  if (true) {  
    var x = 2;  
    console.log(x);  
  }  
  console.log(x);  
}
```

```
function letTest() {  
  let x = 1;  
  if (true) {  
    let x = 2;  
    console.log(x);  
  }  
  console.log(x);  
}
```

Sans mode strict

```
function foo() {  
    var a = b = 5;  
    console.log(a);  
    console.log(b);  
}  
foo();  
console.log(a);  
console.log(b);
```

Mode strict

```
"use strict";
```

```
function foo() {  
    var a = b = 5;  
}  
foo();  
console.log(a);  
console.log(b);
```

Hoisting (hissage)

Que va afficher l'extrait suivant ?

```
function test() {  
  console.log(a);  
  console.log(foo());
```

```
  var a = 1;  
  function foo() {  
    return 2;  
  }
```

```
}  
test();
```

Et avec let au lieu de var ?

Point-Virgule

Les points virgules à la fin de chaque ligne ne sont pas obligatoires.
Il est conseillé de les ajouter systématiquement.

```
// Good
return {
  javascript : "fantastic"
};
```

```
// Bad!
return
{
  javascript : "fantastic"
};
```

'Falsiness'

En Javascript, certaines valeurs non-boléennes peuvent être considérée comme fausses:

- 0
- '' (chaîne vide)
- undefined
- null
- NaN

'Falsiness' (usage)

Les autres valeurs sont considérées comme vraies:

```
> let p = { x: 12, y: 42 };  
> let q = undefined;  
> function log( obj, prop ) {  
  if( obj ) {  
    console.log( obj[prop] );  
  }  
}  
> log( p, "x" )  
12  
undefined  
> log( q, "x" )  
undefined
```

'Falsiness': Exemples

```
> if( 1 ) { console.log("vrai"); }
```

```
vrai
```

```
undefined
```

```
> if( 2 ) { console.log("vrai"); }
```

```
vrai
```

```
undefined
```

```
> if( 0 ) { console.log("vrai"); }
```

```
undefined
```

```
> 0 == false
```

```
true
```

```
> 1 == true
```

```
true
```

```
> 2 == true
```

```
false
```

Opérateurs de comparaison

Il est conseillé d'utiliser l'opérateur `===` à la place de l'opérateur `==` pour les comparaisons:

```
> 0 === false
```

```
false
```

```
> 1 === true
```

```
false
```

```
> 2 === false
```

```
false
```

```
> null == undefined
```

```
true
```

```
> null === undefined
```

```
false
```

Triple égale: Exemple

```
> let p = { x: 12, y: 42 };  
> let q = undefined;  
> function log( obj, prop ) {  
  if( obj !== undefined ) {  
    console.log( obj[prop] );  
  }  
}  
> log( p, "x" );  
12  
undefined  
> log( q, "x" );  
undefined
```

Structures de contrôle

Les structures de contrôle JS sont similaires à celle du langage C. Il est conseillé de n'utiliser que:

- `if`
- `while`

Les structures suivantes existent mais sont déconseillées:

- `for` devenue peu utile à partir d'ES5
- `with` source de nombreux bugs et interdite en mode stricte
- `switch` source de nombreux bugs à cause du *fall-through*.
- `try...catch` les exceptions rendent le code plus complexes.